

SQL::Interpolate – request for comments

Source: <http://coding.derkeiler.com/Archive/Perl/perl.dbi.users/2003-12/0239.html>

From: David Manura (dm.list_at_math2.org)

Date: 12/27/03

Date: Fri, 26 Dec 2003 22:39:35 -0500

To: dbi-users@perl.org

Details are given below on a simple Perl module I wrote called SQL::Interpolate. I'm interested if others find this useful, find it redundant to existing modules, or have better ideas for designing it, as I'm considering submitting it to CPAN. (And if this is not the best place to propose this question, please let me know.) This module arose as I was writing a lot of SQL construction/variable binding code (similar to Recipe 14.13—Building Queries Programmatically, Perl Cookbook, 2nd ed., Christiansen & Torkington) that was able to be more simply expressed using a "dbi_interpolate" function as shown. I had previously tried using SQL::Abstract for this purpose and with some success, but it seemed less expressive than SQL (lacking certain constructs), failed under certain cases (e.g. using 'IN' with a zero-length list), and its syntax for nested AND/OR expressions wasn't as intuitive as in plain SQL.

=====

web page: <http://www.math2.org/david/sql-interpolate/>

current version: 0.1

=====

ABSTRACT

The purpose of SQL::Interpolate is to make writing SQL queries in Perl more natural, less redundant, and less error-prone. SQL::Interpolate takes your query specification and generates a correctly formatted SQL statement along with a list of bind values. These result values can then be passed to DBI or used for another purpose. SQL::Interpolate serves a purpose similar to that of SQL::Abstract <http://search.cpan.org/%7Enwiger/SQL-Abstract/Abstract.pm> except that SQL::Interpolate still exposes and utilizes the full native SQL syntax of your database.

SYNOPSIS

```
use DBI;
use SQL::Interpolate;

my $rows = $dbh->selectall_arrayref(dbi_interpolate qq[
```

```
SELECT * FROM table
WHERE color IN], \@colors, qq[
    AND y = ], \$x, qq[
LIMIT], [1, 10]
);
```

```
$dbh->do(dbi_interpolate qq[
    INSERT INTO table ], {
    color => $new_color,
    shape => $new_shape}
);
```

```
$dbh->do(dbi_interpolate qq[
    UPDATE
```